EXHIBIT 1

CMP
United Business Media

BYTE.com

BYTE ARTICLES BYTEMARKS FACTS

Search BYTE.com

Search

Write to Byte
Editorial Calendar

Categories
Previous Editions
Columns
Features

Print Archives
1994-1998

About Us
Byte Editorial Staff
Advertise with Byte
Privacy Policy

ARTICLES

Building the Virtual PC

November 1997 / Core Technologies / Building the Virtual PC

A software emulator shows that the PowerPC car emulate another computer, down to its very hardv

Eric Traut

Development of Virtual PC -- Connectix Corporation's Macintosh applicati emulates a PC and its peripherals -- began almost two years ago, in October The goal from the beginning was to create a fully Intel-compatible PC in so: The effort centered around a core Pentium instruction-set emulator, complet MMX instructions. True PC emulation also required the reverse-engineerin development of a dozen other PC motherboard devices, including modern peripherals such as an accelerated SVGA card, an Ethernet controller, a Sou Blaster Pro sound card, IDE/ATAPI controller, and PCI bridge interface. Th strategy of hardware-level emulation resulted in an application that allows Macintosh users to run not only Windows programs and DOS games but se x86-based OSes, including Windows 95, NT, and NeXT OpenStep.

Pentium Emulation

The heart of Virtual PC is the Pentium recompiling emulator, a sophisticate of software written entirely in hand-coded PowerPC assembly language. Its translate Pentium instruction sequences into a set of optimized PowerPC instructions that perform the same operation. Translation occurs on a "basic basis, where a basic block consists of a sequence of decoded x86 instruction blocks end on an instruction that abruptly changes the flow of execution (ty jump, call, or return-from-subroutine instruction). As the recompiler decode instructions, it analyzes them for "condition code" u sage. Finally, it generat block of PowerPC code that accomplishes the same task. For more details o process, see "Virtual PC Operation" .

For purposes of speeding things up, the emulator employs the following tric

Translation cache: Even though written in PowerPC assembly language, tl translator still requires substantial time to generate optimized instruction translations. To reduce this overhead, the emulator caches blocks of translat

Interinstruction optimization: Because the Pentium is a CISC processor, ı instructions perform more than one operation. For example, the ADD instru not only adds two values together, it also produces a number of condition- c flags that tell programs whether the addition produced a zero or negative res Such codes are used, for example, to determine if a program performs a con jump. Most of the time these codes are ignored. The translator analyzes bloc x86 instructions to dete rmine which flags the program uses (if any). It then generates PowerPC code for those flags actually used. The first two listings "Translated Code" show how one Pentium instruction translates into three PowerPC instructions, while three Pentium instructions can be optimized fr into five PowerPC instructions.

Address translation: One of the most difficult Pentium features to emulate built-in memory management unit (MMU). This hardware translates linear (logical) addresses into physical memory addresses. Operating systems use tl MMU to implement virtual memory and memory protection. Because of the Pentium's small register file, about three in four Pentium instructions referei memory in one way or another. Each memory address potentially needs to b translated before the emulator loads from, or stores to, the referenced addres MMU implemented in software would impose a high overhead, which woul degrade performance. Luckily, this overhead can be avoi ded: The Connecti engineers were able to program the PowerPC's MMU to mimic the Pentium MMU's behavior, thus managing the address translations in hardware. The Pentium's memory page attributes can also be mirrored in the PowerPC's M For example, if Virtual PC's emulated OS marks a memory page as write-pr the page mappings are modified so the corresponding PowerPC page is writ protected.

Segment bounds checking: The Pentium architecture includes the archaic ı of memory segments. Every memory reference, such as instruction fetches, operations, loads, and stores, has an associated memory segment. When a segment's bounds are exceeded, the Pentium's MMU generates a general pro fault (GPF). The OS uses GPFs for more than detecting bugs in applications enable a program to "thunk" down into privileged driver-level code not acce at the application level. Therefore, the Pentium emulator must detect segme bound faults where appropriate. Although the PowerPC does not contain segmentation hardware akin to the Pentium, Connectix used PowerPC `trap` instructions to perform segment bounds checks with little or no overhead.

Hardware Emulation

Besides the Pentium processor, a typical PC motherboard contains a dozen (chips that work together concurrently. All these chips need to be emulated faithfully for compatibility. The Intel architecture provides an I/O address sı that's used to access hardware outside of the CPU. You work with this "I/O through two instructions -- `IN` and `OUT`. When using these instructions, softv must specify an I/O port (or address). Virtual PC routes I/O accesses to code modules that emulate each chip. For example, if Virtual PC encounters an I instruction referencing port 0x21, it calls a routine in the interrupt-controllei emulation module that returns the current interrupt mask. Similar module ca occur for every I/O space access, as the third listing in "Translated Code" sh

Many of the extra chips on a PC motherboard control I/O devices such as th drive, CD-ROM, keyboard, and mouse. For compatibility with the Mac OS Macintosh hardware, Virtual PC performs all I/O through the standard Mac drivers. So, a request sent to the emulated PC's IDE controller to read a sect the hard drive gets translated into a read operation that's sent to the Mac OS driver.

The most difficult hardware components to emulate involve precise timing. example, sound is a real-time operation, and any timing perturbation results clicks or pops as digitally sampled data fails to arrive on time. Because Virt is hosted on the Mac OS (which gives time to other Mac programs running concurrently, as well as Virtual PC), and it needs to emulate several dozen I chips in parallel, precise timing isn't always possible. Virtual PC compensat placing the highest priority on tasks that directly affect the user, such as sou video.

Performa nce

Emulated systems are naturally going to be slower than real hardware. But Connectix engineers concentrated on tuning aspects of the emulated hardwa required to run popular PC games and productivity applications at a usable performance level. This was especially challenging given that the PowerPC processor emulates not only the Pentium but all the other chips on a PC motherboard.

Performance of Virtual PC is also greatly affected by the host hardware syst The latest PowerPC processors with high clock rates and large on-chip cach run it best. The speed and size of the system's L2 cache is also critical becau the code expansion that occurs during the translation process.

While users will take a performance hit because this is an emulator, Virtual successfully emulates the entire PC at a very low level. PC programs -- applications, device drivers, and operating systems alike -- cannot tell they a running on actual PC hardware.

Translate d Code

Translation of Single Pentium Instruction

```
Pentium instruction               PowerPC instructions

ADD EAX,20                        li          rTemp1,20
                                  addco.      PF,rTemp1,rEA
                                  mr          rEAX,rPF
```

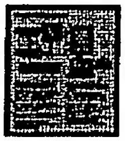
Translation of Pentium Instruction Block

```
Pentium instructions              PowerPC instructions

ADD EAX,20                        add           rEAX,rEAX,20
ADD EBX,30                        add           rEBX,rEBX,30
ADD ECX,40                        li            rTemp1,40
                                  addco.        rPF,rTemp1,rE
                                  mr            rECX,rPF
```

Code Translation for Pentium I/O Instructions

```
Pentium instructions              PowerPC instructions

MOV AL,8                          li            rAL,8
MOV DX,0x1F0                      li            rDX,0x1F0
OUT DX,AL                         bl            HandleIDEPort
AD
D DX,7                          addi          rDX,rDX,7
IN AL,DX                          bl            HandleIDEPort
RET                               addi          rIP,rIP,8
                                  b             DispatchToNe>
```

Virtual PC Operation

illustration link (24 Kbytes) 

Eric Traut (mailto:traut@connectix.com) is lead engineer for Virtual PC at Con Apple Computer, he wrote the 680x0 dynamic recompiling emulator for PowerP(

Up Level Previous Next

BYTE TOP OF PAGE